FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA XVII.

Kolozsvár, 2012. március 22–23.

GRAFIKUS KÁRTYA SZÁMÍTÁSI KAPACITÁSÁNAK HASZNÁLATA DETERMINISZTIKUS, EGYKÉPES ÜTEMEZÉSI PROBLÉMÁK MEGOLDÁSÁRA

MIHÁLY Krisztián, HORNYÁK Olivér

Abstract

This paper gives a short summary of the first results of our research project, where the the graphical processing units is used in non-graphical related problems. First the short history of GPU is summarized with the focus on the calculation capacity and the programming approaches. A brief description is given about the actually available hardware possibilities and the available programming languages. As an initial result an easy and well known scheduling algorithm was implemented for deterministic, single machine models. To check the performance achievements both the CPU and GPU code were implemented and compared. At the end some of our performance measurements will be presented.

Key words:

GPGPU, CUDA, OpenCL, Scheduling Problems

Összefoglalás

A grafikus kártya elsődleges feladata, hogy számítási teljesítményével a központi számító egységet mentesítse a grafikus számítási feladatok alól, úgy mint a 3 dimenziós alakzatokat transzformálásától, láthatósági számításokat elvégzésétől, fény-árnyék megjelenítéseket számoljon és textúrákat illesszen a felületekre. Történelmi és hardver fejlesztési okokból a grafikus kártyák fizikai felépítését is ennek megfelelően optimalizálták. A technológia folyamatos fejlődésével azonban lehetővé vált, hogy az eszköz által nyújtott számítási kapacitást ne csak grafikus, hanem más, általános problémák megoldására is használni lehessen. Kialakult egy új paradigma, amelyet a grafikus processzor általános célú felhasználásának nevezünk (General-Purpose computing on Graphics-Processing Units, GPGPU). Ez a fejlődés ma is tart, a GPGPU programozás a tudományos érdeklődésnek ma is természetes célpontja, hiszen segítségével tudományos kutatási, általános mérnöki alkalmazásokat lehet kidolgozni, vagy meglévő megoldásokat optimalizálni. Cikkünkben bemutatunk egy egyszerű alkalmazási lehetőséget, ahol egy egygépes, determinisztikus, ütemezési problémát oldottunk meg GPU segítségével.

Kulcsszavaj:

GPGPU, CUDA, OpenCL, Ütemezési Problémák

1. A grafikus kártyák fejlődésének történelmi áttekintése

A számítógépek teljesítményének növekedése egészen az ezredfordulóig abban nyilvánult meg, hogy a központi feldolgozó egység (CPU – Central Processing Unit) belső órájának sebessége milyen mértékben növekedett. A legszembetűnőbb példa, hogy az 1980-as években a CPU belső órajele kb. 1 MHz-t tett ki, míg 2000-es években mindez 1 és 4 GHz között mozgott. Ennek a fejlődésnek azonban gátat szabott a használt technológia fizikai korlátai, úgymint a tranzistorok mérete, a hűtési gondok,

illetve maga a fénysebesség. 2005 magasságában ezért a hardvergyártók elkezdtek a több magos processzorok gyártását, amelyet sokan a „többmagos forradalomnak” is neveztek [1]. A CPU fejlődésével egy időben a grafikus kártyák és az azokon található grafikus feldolgozó egységek (GPU – Graphical Processing Unit) is drasztikusnak mondható fejlődésen mentek keresztül. Az 1980-as évek végén és az 1990-es évek elején a grafikus operációs rendszerek közkedveltsége miatt (pl. Windows) a személyi számítógépekben megjelentek a kétdimenziós gyorsítók. Ezek a kétdimenziós gyorsítók megjelenítéssel kapcsolatos számítási feladatokat vettek át a CPU-tól, így a CPU-t más, nem megjelenítés releváns számításokra lehetett használni.

1992-ben a Silicon Graphics nevű cég nyitotta tette termékeihez tartozó programozási felületet, az OpenGL könyvtárakat. Ez standardizált, platform-független lehetőséget adott háromdimenziós alkalmazások fejlesztéséhez. A grafikus cső (pipeline) lépésekre osztja a képszintézist, de ezen az architektúrán a grafikus kártya működésébe még nem lehetett saját implementációt illeszteni. A következő áttörést a Microsoft DirectX 8.0 megjelenése hozta, amely bevezette a programozható csúcs árnyalót (vertex shader) és pikszel árnyalót (pixel shader). Ezt tekinthetjük az első pontnak, amikor az alkalmazásfejlesztőknek lehetőségük nyílt a GPU program futását módosítani. Ez a megközelítés nagyon jó lehetőséget biztosított grafikai alkalmazások fejlesztéséhez, azonban nehézkesnek bizonyult más területeken. Az „adat” ilyenkor egy, a GPU-ra töltött kép formájában értelmezhető, melyen más szín és textúra információk alapján végzünk műveleteket, majd az eredményt egy képként tesszük elérhetővé a központi egységen. Az GPU-n futtatott algoritmus nem ér el memóriát, az adatokat csak közvetett formában használhatja, alaposan ismerni kellett a grafikus eszköz működését, hogy a hardveresen elvégzett számításokat értelmezni lehessen.

A grafikus kártyák új generációja lehetővé teszi, hogy úgy nevezett kerneleket fejlesszünk, mely kernelek a grafikus kártyán, sok magon, párhuzamosan futva, különböző memóriaterületeket elérve tudnak futni. Ezek a kernelek már függetlenek a grafikus pipeline-tól, programozásuk sokkal egyszerűbb. Megoldásunkban az NVidia cég által fejlesztett és elérhetővé tett CUDA programnyelvet használjuk. Létezik más gyártó által készített hasonló platform (Stream), illetve gyártó független standard is (OpenCL), amelyre ebben a cikkben nem térünk ki.

2. A determinisztikus, egygépes, előzésmentes ütemezési modell

Az egygépes, determinisztikus ütemezési modellben definiáljuk a munkák halmazát. Ezt halmazt jelölje J , amely n darab munkát tartalmaz $J = \{j_1, j_2, \dots, j_n\}$. Minden munka rendelkezik egy fontosságot kifejező faktorral (w_j), valamint a feldolgozáshoz szükséges idővel (p_j). Minden munka rendelkezhet egy teljesítési időponttal (C_j), amely azt az időpontot határozza meg, amikor a munka elkészült. A modellben egy gép áll rendelkezésre, amely egy adott munkát p_j idő alatt dolgoz fel. A

munka feldolgozás közben nem szakítható meg. A legegyszerűbb esetben a munkák között nincs sorrendiségi reláció definiálva.

Egy ütemezésnek nevezzük a munkák egy sorrendjét. A különböző lehetséges ütemezések száma ebben a modellben $n!$. Az ütemezési feladat célja, hogy ütemezési célfüggvény(ek) vizsgálatával a megvalósítható legjobb ütemezést megtaláljuk. Az általunk vizsgált modellben a célfüggvény az, hogy az összes, súlyozott teljesítési időt minimalizáljuk. Ezt az irodalomban TWCT célfüggvénynek nevezik (Total Weighted Completion Time), tehát a feladat

$$\min(TWCT), \text{ ahol,} \quad (1)$$

$$TWCT = \sum_{j=1}^n w_j C_j$$

Bizonyított tétel [2], hogy ilyen esetben a „legkisebb súlyozott feldolgozási idővel rendelkezőt először” szabály optimális ütemezést ad.

3. Eredmények

Elsődleges feladat a párhuzamosítási pontok megtalálása. Mivel ebben a modellben a munkák egymástól függetlenül végrehajthatóak, ezért a j_i munkát jellemző w_i/p_i érték is más munkáktól függetlenül számítható. A mérések elvégzéséhez két algoritmust implementáltunk, egyet az egymagos CPU-ra, a másikat a több, párhuzamosan futtatható maggal rendelkező GPU-ra. A CPU algoritmus szekvenciálisan dolgozza fel a munkák állományát, egyesével kiszámítva a w_i/p_i értékeket. A GPU-ra írt algoritmus esetén az adatot előbb az eszközre kell tölteni a futtatandó kernellel együtt, majd a feldolgozást a lehető legtöbb párhuzamos szállal végrehajtani és az eredményt a gazda programba visszaszámolni. Mindkét esetben a w_i/p_i értékek számítása után már csak a munkák sorrendezését kell elvégeznünk.

1. ábra Mérési eredmények

Az algoritmusok részletes leírása a [3] olvasható részletesen. Ebben a cikkben csak mérési eredményeket ismertetjük, melyet az 1.ábrán mutatunk be.

Az ábrán három mérési eredmény található. A CPU-val jelölt eredmény a CPU-n található szekvenciális feldolgozás eredménye, a CUDA 1 azt az esetet mutatja, amikor a mérés során az általános adatmodellből kiolvasott adatot a GPU adatmodelljére képezzük a gazdagépen, a CUDA 2 mérési sorozat eredményében csak az adatok másolását, feldolgozását és visszaolvasását mértük. Látható, hogy amennyiben az adatok „GPU helyes” módon állnak rendelkezésre, akkor a GPU gyorsabb feldolgozási sebességet biztosít.

Köszönetnyilvánítás

A bemutatott kutató munka a TÁMOP-4.2.1.B-10/2/KONV-2010-0001 jelű projekt részeként az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Irodalomjegyzék

- [1] Jasen Sanders, Edward Kandrot: CUDA by example, Addison-Wesley, 2010, 3. oldal
- [2] M. L. Pinedo: Scheduling: Theory, Algorithms, and Systems, Springer, 2008, 36. oldal
- [3] Krisztián Mihály, Olivér Hornyák: Using Graphical Processing Unit in Deterministic Single Machine Scheduling Problems, Submitted to Production Systems and Information Engineering, Volume 7 (2012),

Mihály Krisztián, Doktorandusz

SAP Hungary Kft.

Budapest 1037 Záhony utca 7.

Telefon: (36) 1 885 7267 , (36) 70 377 0162

Dr. Hornyák Olivér, PhD

Miskolci Egyetem, Alkalmazott Informatika Tanszék, Miskolc-Egyetemváros 3515

Telefon: (36) 46-565-111 / 19-54